School of Data and Computer Science, Sun Yat-sen University

E04 Futoshiki Puzzle (Forward Checking)

16110917 Zhaoshuai Liu

September 28, 2018

Contents

1 Futoshiki

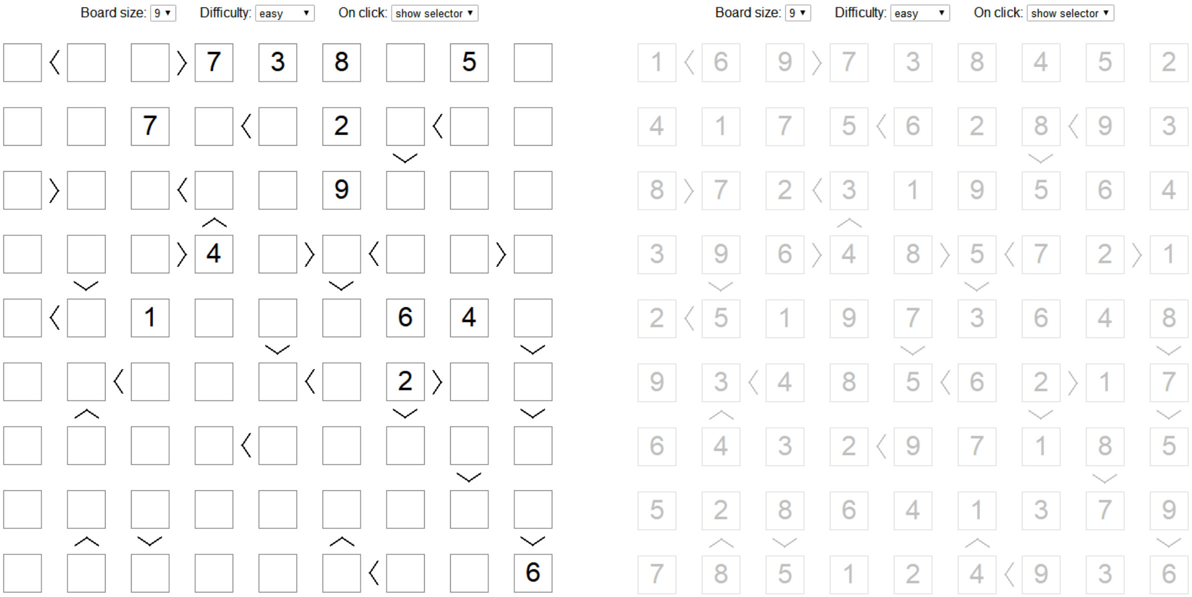
Futoshiki is a board-based puzzle game, also known under the name Unequal. It is playable on a square board having a given fixed size (4 × 4 for example).

The purpose of the game is to discover the digits hidden inside the board's cells; each cell is filled with a digit between 1 and the board's size. On each row and column each digit appears exactly once; therefore, when revealed, the digits of the board form a so-called Latin square.

At the beginning of the game some digits might be revealed. The board might also contain some inequalities between the board cells; these inequalities must be respected and can be used as clues in order to discover the remaining hidden digits.

Each puzzle is guaranteed to have a solution and only one.

You can play this game online: `http://www.futoshiki.org/`.



Board size: 9 Difficulty: easy On click: show selector

```
□ < □   □ > 7   3   8   □   5   □
□   □   7   □ < □   2   □ < □   □
                        v
□ > □   □ < □   □   9   □   □   □
            ^
□   □   □ > 4   □ > □ < □   □ > □
    v               v
□ < □   1   □   □   □   6   4   □
                v               v
□   □ < □   □   □ < □   2 > □   □
    ^                   v       v
□   □   □   □ < □   □   □   □   □
                            v
□   □   □   □   □   □   □   □   □
    ^   v           ^           v
□   □   □   □   □   □ < □   □   6
```

Board size: 9 Difficulty: easy On click: show selector

```
1 < 6   9 > 7   3   8   4   5   2
4   1   7   5 < 6   2   8 < 9   3
                        v
8 > 7   2 < 3   1   9   5   6   4
            ^
3   9   6 > 4   8 > 5 < 7   2 > 1
    v               v
2 < 5   1   9   7   3   6   4   8
                v               v
9   3 < 4   8   5 < 6   2 > 1   7
    ^                   v       v
6   4   3   2 < 9   7   1   8   5
                            v
5   2   8   6   4   1   3   7   9
    ^   v           ^           v
7   8   5   1   2   4 < 9   3   6
```

Figure 1: An Futoshiki Puzzle

2 Tasks

1. Please solve the above Futoshiki puzzle (Figure 1) with forward checking algorithm.

2. Write the related codes and take a screenshot of the running results in the file named E04_YourNumber.pdf, and send it to ai_2018@foxmail.com.

3 Codes

```
#include<iostream>
#include<vector>
#include<set>
#include <algorithm>
using namespace std;

bool assigned(set<int> unassigned)
{
    return unassigned.empty();
}

int pickUnassigned(set<int> unassigned)
{
    return *(unassigned.begin());
}

bool FCCheck(int pos, int table[][9], bool dom_row[][9], bool dom_col[][9],
vector<pair<int, int> > con)
{
    int x = pos / 9;
    int y = pos % 9;

    set<int> dom_cur;//domain for (x, y)

    int i;
    for(i = 0;i < 9;i++)//row_dom for (x, y)
        if(dom_row[x][i] && dom_col[y][i])
            dom_cur.insert(i + 1);

    vector<pair<int, int> >::iterator is;
    set<int >::iterator it;
```

```
for(is = con.begin(); is != con.end(); is++)
{
    int other, y_col, y_row;
    other = -1;
    if((*is).first == pos)
    {
        other = (*is).second;
        y_row = other/9;
        y_col = other%9;
        if(table[y_row][y_col] != 0)
        {
            for(it = dom_cur.begin(); it != dom_cur.end(); it++)
            {
                if(*it > table[y_row][y_col])
                    dom_cur.erase(it);
            }
        }
    }
    else if((*is).second == pos)
    {
        other = (*is).first;
        y_row = other/9;
        y_col = other%9;
        if(table[y_row][y_col] != 0)
        {
            for(it = dom_cur.begin(); it != dom_cur.end(); it++)
            {
                if(*it < table[y_row][y_col])
                    dom_cur.erase(it);
            }
        }
    }
}
```

```
    if(dom_cur.empty())
        return false;
    else
        return true;
}

void FC(int table[][9], bool dom_row[][9], bool dom_col[][9], vector<pair<int, int
{
    if(assigned(unassigned))
        return;
    int pos = pickUnassigned(unassigned);
    unassigned.erase(pos);
    cout << "pos:␣" << pos << endl;
    int x = pos/9;
    int y = pos%9;
    //fixed[x][y] = true;

    set<int> dom_cur;//domain for (x, y)

    int i;
    for(i = 0;i < 9;i++)//row_dom for (x, y)
        if(dom_row[x][i] && dom_col[y][i])
            dom_cur.insert(i + 1);




    set<int>::iterator it;
    vector<pair<int, int> >::iterator is;
    for(is = con.begin();is != con.end();is++)
    {
        int other, y_col, y_row;
        other = -1;
        if((*is).first == pos)
    {
        other = (*is).second;
        y_row = other/9;
        y_col = other%9;
        if(table[y_row][y_col] != 0)
        {
            for(it = dom_cur.begin(); it != dom_cur.end(); it++)
            {
                if(*it > table[y_row][y_col])
                    dom_cur.erase(it);
            }
        }
    }
    else if((*is).second == pos)
    {
        other = (*is).first;
        y_row = other/9;
        y_col = other%9;
        if(table[y_row][y_col] != 0)
        {
            for(it = dom_cur.begin(); it != dom_cur.end(); it++)
            {
                if(*it < table[y_row][y_col])
                    dom_cur.erase(it);
            }
        }
    }
}
if(dom_cur.empty())
    return;
for(it = dom_cur.begin(); it != dom_cur.end(); it++)
{
    int value = *it;
```

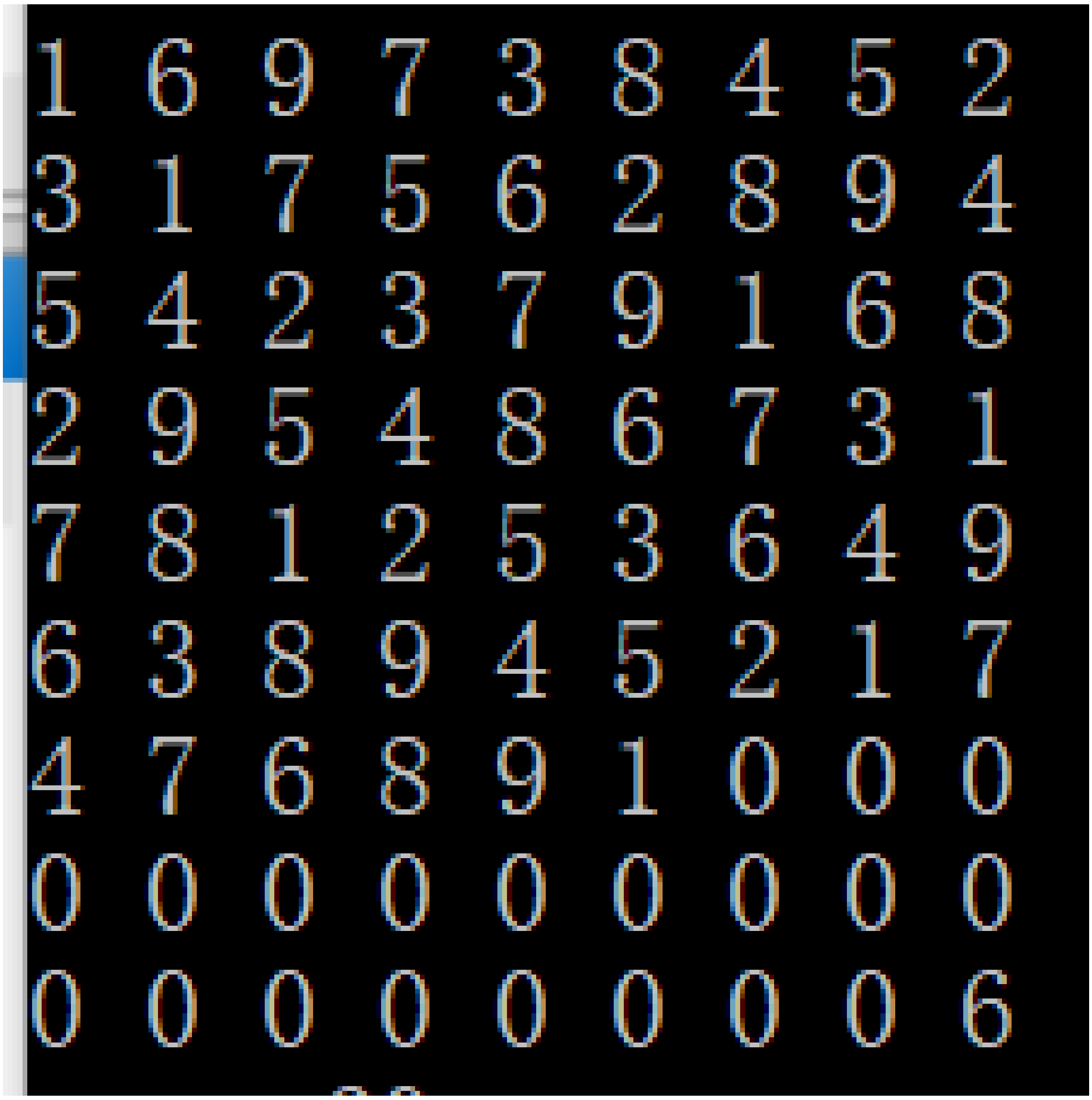
```
cout << "value:␣" << value << endl;
table[x][y] = value;
dom_row[x][value-1] = false;
dom_col[y][value-1] = false;
bool DWOoccured = false;
for(is = con.begin(); is != con.end(); is++)
{
    int other, y_col, y_row;
    other = -1;
    if((*is).first == pos)
    {
        other = (*is).second;
        y_row = other/9;
        y_col = other%9;
        cout << "other:␣" << other << endl;
        if(table[y_row][y_col] == 0)
        {
            if(!(FCCheck(other, table, dom_row, dom_col, con)))
            {
                DWOoccured = true;
                break;
            }
        }
    }
    else if ((*is).second == pos)
    {
        other = (*is).first;
        y_row = other/9;
        y_col = other%9;
        cout << "other:␣" << other << endl;
        if(table[y_row][y_col] == 0)
        {
            if (!(FCCheck((*is).first, table, dom_row, dom_col, con)))
```

```
                    {
                        DWOoccured = true;
                        break;
                    }

                }
            }

        }
        if(!DWOoccured)
        {
            cout << endl;
            int j;
            for (i = 0; i < 9; i++)
            {
                for (j = 0; j < 9; j++)
                    cout << table[i][j] << " ";
                cout << endl;
            }
            FC(table, dom_row, dom_col, con, unassigned);
        }
        dom_row[x][value - 1] = true;
        dom_col[y][value - 1] = true;
        table[x][y] = 0;
    }
    unassigned.insert(pos);
    return ;
}

int main()
{
    int table[9][9];
    bool dom_row[9][9]; //row_domain
bool dom_col[9][9];//col_domain
vector<pair<int, int> > con;//constraint
set<int> unassigned;

int i, j;
for(i = 0;i < 9;i++)
{
    for (j = 0; j < 9; j++)
    {
        table[i][j] = 0;
        //fixed[i][j] = false;
        dom_row[i][j] = true;
        dom_col[i][j] = true;
    }
}
table[0][3] = 7;
table[0][4] = 3;
table[0][5] = 8;
table[0][7] = 5;

table[1][2] = 7;
table[1][5] = 2;

table[2][5] = 9;

table[3][3] = 4;

table[4][2] = 1;
table[4][6] = 6;
table[4][7] = 4;

table[5][6] = 2;
```

```
table[8][8] = 6;

for (i = 0; i < 9; i++)
{
    for (j = 0; j < 9; j++)
        cout << table[i][j] << ”␣”;
    cout << endl;
}

con.push_back(make_pair(0, 1));
con.push_back(make_pair(3, 2));
con.push_back(make_pair(12, 13));
con.push_back(make_pair(15, 16));
con.push_back(make_pair(19, 18));
con.push_back(make_pair(20, 21));
con.push_back(make_pair(24, 15));
con.push_back(make_pair(21, 30));
con.push_back(make_pair(30, 29));
con.push_back(make_pair(37, 28));
con.push_back(make_pair(30, 29));
con.push_back(make_pair(32, 31));
con.push_back(make_pair(32, 33));
con.push_back(make_pair(35, 34));
con.push_back(make_pair(36, 37));
con.push_back(make_pair(37, 28));
con.push_back(make_pair(41, 32));
con.push_back(make_pair(46, 37));
con.push_back(make_pair(46, 55));
con.push_back(make_pair(49, 40));
con.push_back(make_pair(49, 50));
con.push_back(make_pair(52, 51));
con.push_back(make_pair(60, 51));
con.push_back(make_pair(53, 44));
con.push_back(make_pair(46, 55));
con.push_back(make_pair(57, 58));
con.push_back(make_pair(60, 51));
con.push_back(make_pair(62, 53));
con.push_back(make_pair(70, 61));
con.push_back(make_pair(64, 73));
con.push_back(make_pair(74, 65));
con.push_back(make_pair(68, 77));
con.push_back(make_pair(80, 71));
con.push_back(make_pair(77, 78));

dom_row[0][6] = dom_col[3][6] = false;
dom_row[0][2] = dom_col[4][2] = false;
dom_row[0][7] = dom_col[5][7] = false;
dom_row[0][4] = dom_col[7][4] = false;
dom_row[1][6] = dom_col[2][6] = false;
dom_row[1][1] = dom_col[5][1] = false;
dom_row[2][8] = dom_col[5][8] = false;
dom_row[3][3] = dom_col[3][3] = false;
dom_row[4][0] = dom_col[2][0] = false;
dom_row[4][5] = dom_col[6][5] = false;
dom_row[5][1] = dom_col[6][1] = false;
dom_row[8][5] = dom_col[8][5] = false;


for (i = 0; i < 9; i++)
    for (j = 0; j < 9; j++)
        if (table[i][j] == 0)
            unassigned.insert(i * 9 + j);

FC(table, dom_row, dom_col, con, unassigned);

cout << endl;
```

```
    for (i = 0; i < 9; i++)
    {
        for (j = 0; j < 9; j++)
            cout << table[i][j] << ”␣”;
        cout << endl;
    }

    return 0;
}
```

4 Results



```
1 6 9 7 3 8 4 5 2
3 1 7 5 6 2 8 9 4
5 4 2 3 7 9 1 6 8
2 9 5 4 8 6 7 3 1
7 8 1 2 5 3 6 4 9
6 3 8 9 4 5 2 1 7
4 7 6 8 9 1 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 6
```